Problems: Unit 2

Problem 1

We have n natural numbers, where n is an even number, which have to come together forming pairs of two numbers each. Then, from each pair, the sum of its two components is obtained, and from all these results the maximum is taken.
Design a greedy algorithm that creates the pairs so that the maximum value of the sums of the numbers of each pair is as small as possible, showing that the candidate selection function used provides an optimal solution.
Example: assuming that the data is in the following vector

5	8	1	4	7	9

Let's see a couple of ways to solve the problem (not necessarily the optimal one):
We select as a couple the consecutive elements
In this way we get the pairs (5, 8), (1, 4) and (7, 9); then, adding the components we have the values 13, 5 and 16, so the final result is 16.
We select as a couple the opposite elements in the vector
Now we have the pairs (5, 9), (8, 7) and (1, 4); adding we get 14, 15 and 5, so the final result is 15 (better than before).
Will there be a better result for this problem? Can a method be generalized that gives us a correct greedy algorithm for any amount of data, and that is independent of the value of those data?

Problem 2

A set of n files must be stored on a magnetic tape (sequential path storage medium), each file having a known length $l_1, l_2, \ldots, l_n$. To simplify the problem, it can be assumed that the reading speed is constant, as well as the information density in the tape.
The rate of use of each stored file is known in advance, that is, the number of p_i requests corresponding to the file i that will be made is known. Therefore, the total of requests to the support will be the quantity $\boldsymbol{P = \sum_{i=1}^{n} p_i}$. After requesting a file, when it is found, the tape is automatically rewound until its beginning.
The objective is to decide the order in which the n files should be stored so that the average loading time is minimized, creating a correct greedy algorithm.

Problem 3

There is a vector V formed by n data, from which we want to find the minimum element of the vector and the maximum element of the vector. The type of data that is in the vector is not relevant to the problem, but the comparison between two data to see which of them is lower is very expensive, so the algorithm for the search of the minimum and the maximum should make the least amount of comparisons between possible elements.
A trivial method consists of a linear path of the vector to search for the maximum and then another path to search for the minimum, which requires a total of approximately 2n comparisons between data. This method is not fast enough, so it is requested to implement a method with greedy methodology that makes a maximum of $\boldsymbol{\frac{3}{2}n}$ comparisons.

Problem 4

We have a non-directed graph G = <N, A>, where N = {1, ..., n} is the set of nodes and $A \subseteq NxN$ is the set of edges. Each edge $(i, j) \in A$ has an associated cost c_{ij} ($c_{ij} > 0\ \forall i, j \in N$; if $(i, j) \notin A$ can be considered $c_{ij} = +\infty$). Let M be the cost matrix of the graph G, that is, M [i, j] = c_{ij} (since the non-directed graph is that (i, j) = (j, i) so the matrix M is symmetric).

Having as data the number of nodes n and the cost matrix M, it is requested to find the minimum support tree of graph G using Prim's algorithm, using the following ideas:

- Unlike the Kruskal algorithm (which creates the tree using independent connected components that are joined together), Prim's algorithm is based on the idea of building an increasingly large tree, starting with a single node and ending by coating the entire graph.
- The algorithm begins with a tree of a node, to which a second node is added, then a third node, etc., until the n nodes are joined. The way to choose a node is looking for the nearest node to the whole tree, without creating cycles.
- As the size of the tree grows, the search for the nearest node becomes complicated, so for the algorithm to be efficient (the method must have $O(n^2)$) a data structure must be created that stores the best distance from each node to the set of nodes of the tree.
- It will be necessary to store in some way the way in which the tree has been created, for example by indicating to which node of the tree the new selected candidate is joining.

Problem 5

We have a directed graph G = <N, A>, where N = {1, ..., n} is the set of nodes and $A \subseteq NxN$ is the set of edges. Each edge $(i, j) \in A$ has an associated cost c_{ij} ($c_{ij} > 0\ \forall i, j \in N$; if $(i, j) \notin A$ can be considered $c_{ij} = +\infty$). Let M be the cost matrix of the graph G, that is, M [i, j] = c_{ij}.

Taking as data the number of nodes n and the cost matrix M, it is requested to find both the minimum path between nodes 1 and n and the length of said path using the Dijkstra algorithm, using the following ideas:

- Create a data structure that stores the known temporal distances (initialized at the cost of the edge of the vertex 1 to each vertex j, or $+\infty$ if that edge does not exist) for the vertices not traveled (initially, all but 1).
- Select as candidate the one with the least known temporal distance, eliminate it from the set of vertices not traveled, and update the rest of the temporal distances if they can be improved using the current vertex.
- It will be necessary to store the way of traversing the graph from vertex 1 to vertex n (not necessarily equal to the set of decisions taken).

Problem 6

Shrek, Donkey and Dragona arrive at the foot of Lord Farquaad's towering castle to free Fiona from her confinement. As they suspected that the drawbridge would be guarded by numerous soldiers, many stairs have been brought, of different heights, with the hope that some of them will allow them to overcome the wall; but no stairs serve them because the wall is very high. Shrek realizes that if he could combine all the stairs into one, he would be able to get to the top exactly and be able to enter the castle.

Fortunately the stairs are made of iron, so with the help of Dragona they go to "weld" them. Dragona can weld any two stairs with her fire breath, but it takes to heat the extremes as many minutes as meters add up the stairs to be welded. For example, welding

6 and 8 meters of stairs would take 6+8=14 minutes. If this ladder was then welded to a 7-meter ladder, the new time would be 14 +7=21 minutes, so it would have taken a total of 14+21=35 minutes to complete the ladder.
Design an efficient algorithm that finds the best cost and way to weld the stairs so that Shrek takes as little time as possible to scale the wall, indicating the chosen data structures and their way of use. It can be assumed that exactly the stairs necessary to climb the wall are available (neither left nor missing), that is, the data of the problem is the collection of measurements of the "mini-ladders" (in the data structure chosen), and that only the optimal way of melting the stairs is looked for.

Deliverables: An exercise to choose among problems 2 and 3, an exercise to choose among problems 4 and 5 and problem 6.